What is Git?

Git is a version control tool for software development. To put it simply, code changes a lot, and Git helps developers track, monitor, and manage those changes.

What is GitHub?

GitHub is a platform where developers can collaborate and do version control on their software. When you get the hang of it, it'll be easier to work together on group projects. You won't need to use Google Drive or send code using Facebook Messenger.

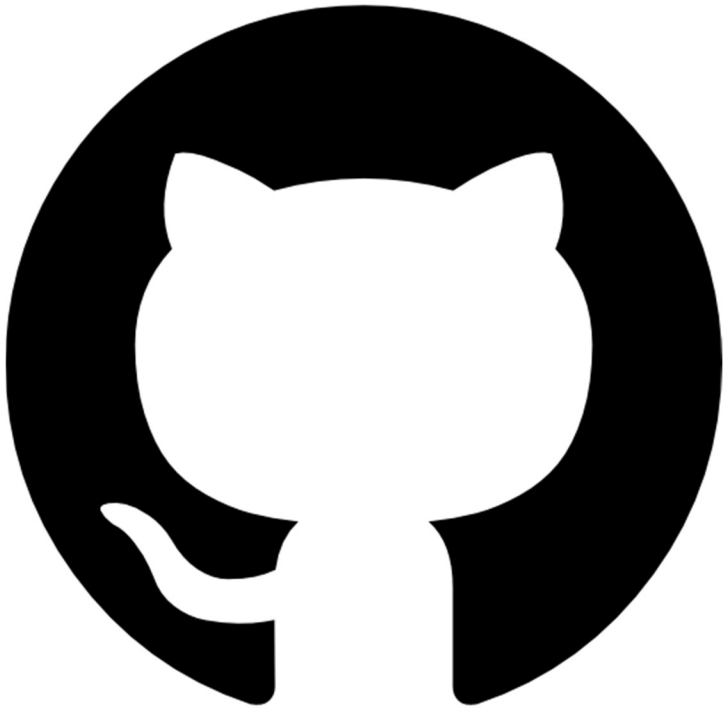
For context, here are some things GitHub has helped me with:

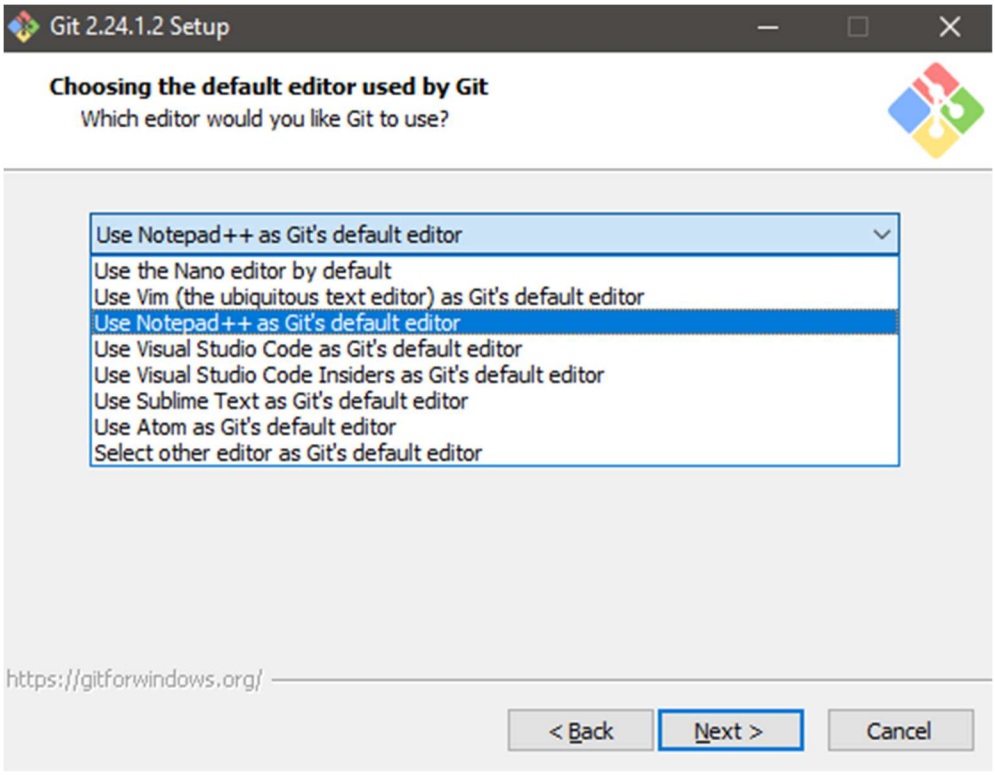
1. A feature in our group project suddenly broke, but we were able to look at our code history and fix what broke it.
2. It makes it easier to keep track of the contribution of my group mates. This is especially helpful when you need a way to quantify contribution per member.
3. Do you ever feel afraid to change code in your project? I mess up a lot, but I can easily reset any changes I make. I also work on a separate "copy" of a project when I'm adding a new feature.

Installing Git

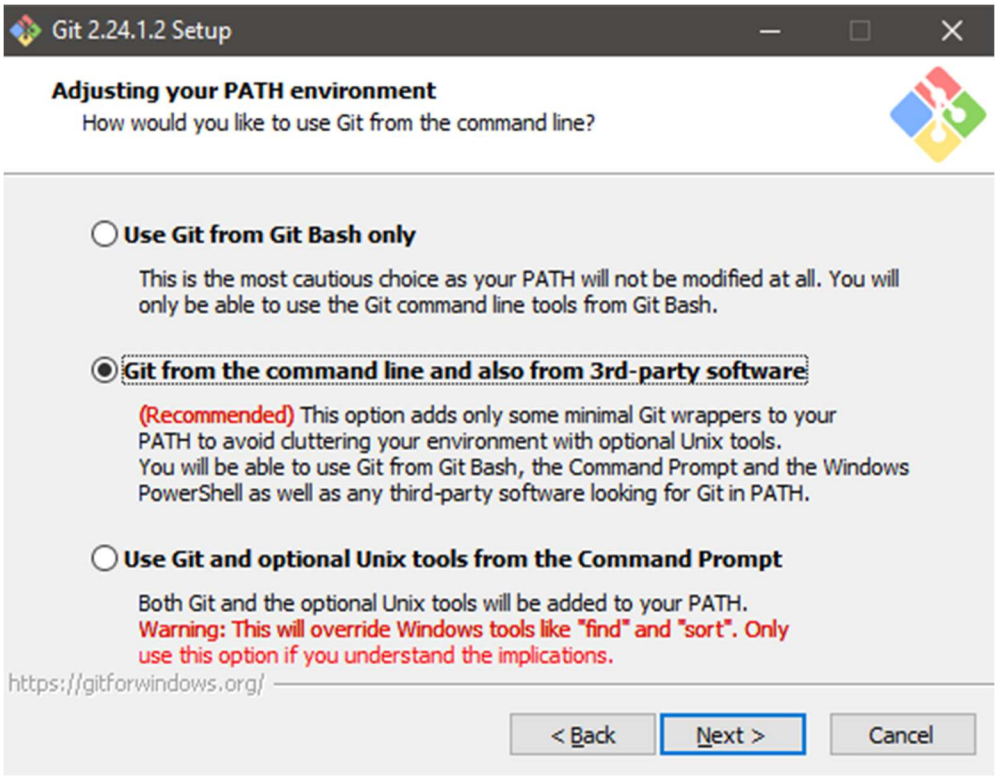
To install Git on Windows, go to https://git-scm.com/download/win.

You can keep most of the default settings, but I recommend changing the default editor to something you're more comfortable with:

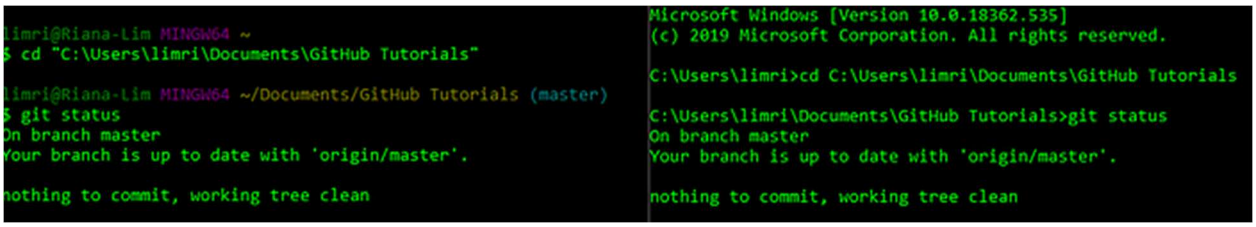




And also allow using Git from the command line, if you prefer using this over Git Bash.



Git Bash is kind of like a command line, but for Git. Here's a side by side comparison of the two.



Git Bash on the left, Windows CMD on the right

After installing, open your command line (or bash) and run git config --global user.name <your name> to set up your Git username. Adding --global will apply this email address of the user of your system. If you just want to set-up the email for the specific repository you're working on, you can change this to --local.

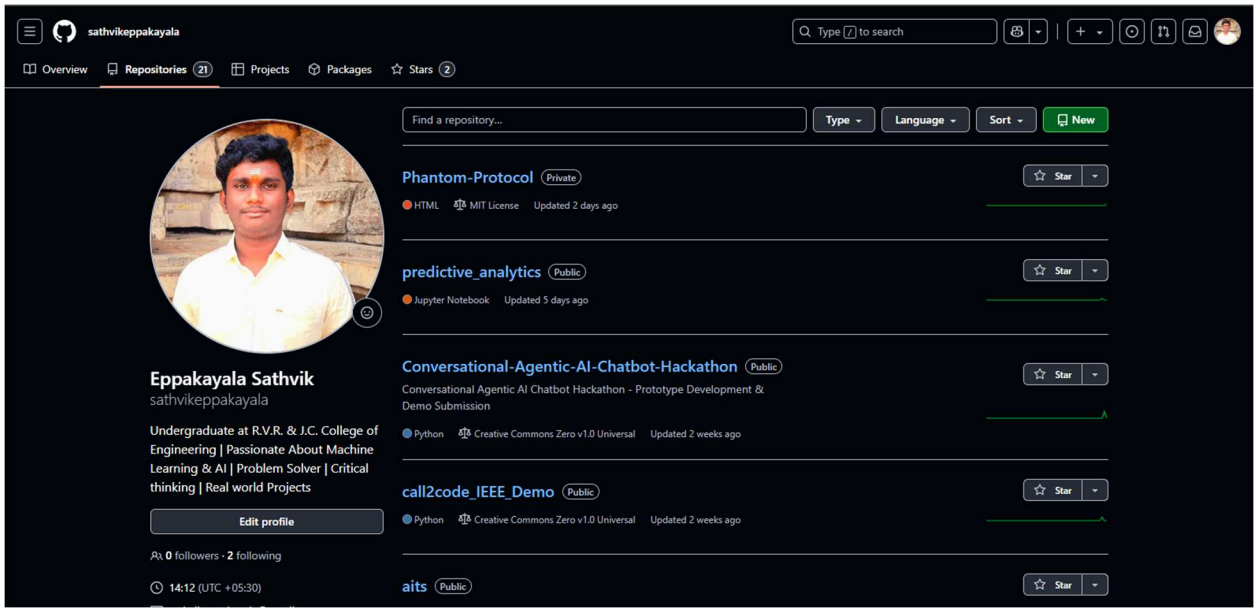
After setting up your Git username, run git config --global user.email "your_email@example.com" to set-up Git to use your email.

Creating a Repository on GitHub

I will assume that you've created an account on GitHub (it isn't any different from creating an account on other websites). **If you are from the Ateneo, you can add your OBF email to your account for a pro account.**

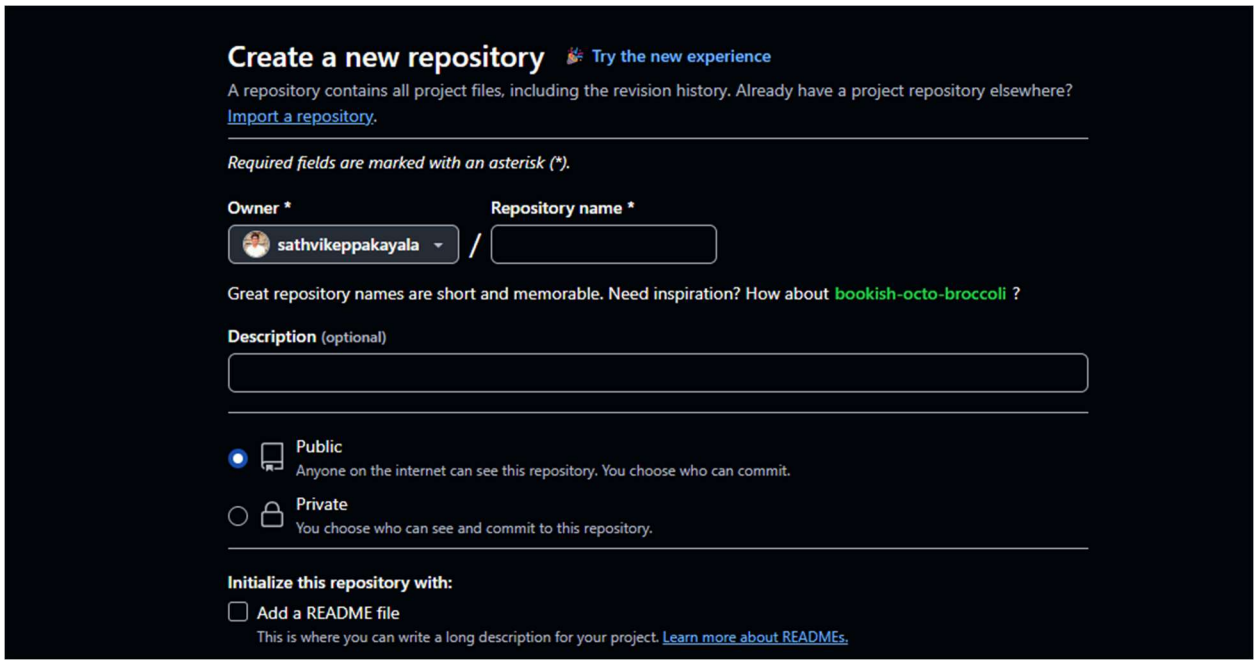
In GitHub, projects are called *repositories*. You can think of repositories as "folders" where all your project files are. The repository is where you do all the version control stuff.

First, you need to create a repository. You can find the "New " button above your list of repositories in the home page, or when you navigate to your profile and click on "Repositories".



Look out for a green button that says “New”

You’ll be taken to this page:



There are two possible ways to move forward depending on what you want to do:

1. Pushing into GitHub repository

Since you're moving a project you already have into a GitHub repository, don't tick the box that says "Initialize this repository with a **README**".

You'll be taken to a page that looks like this:

To turn a project you already have into a repository, open the terminal and move to your project's folder. To do this, run the command cd <directory address>. (CD means Change Directory)

```
rvr@DESKTOP-DI48LBN MINGW64 ~ (master)
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   restore    Restore working tree files
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   diff       Show changes between commits, commit and working tree, etc
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status
```

Run the command git init. This will *init*ialize a Git repository.

```
C:\Users\limri\Documents\GitHub Tutorials>git init
Initialized empty Git repository in C:/Users/limri/Documents/GitHub Tutorials/.git/
```

Run the command git add . The "." means that you're adding all the created, changed, or deleted files that are in your folder to the commit you're making. You can think of a commit as like save states in video games. You changed something in your repository, so you wanna save the state it's in so you could go back to it later on if you need to. **Note that it is bad practice to add all files at once, but since we are learning how to create repositories first we can let it slide for now.**

Next, run the command git commit -m "<your commit message>". Let's dissect that for a bit. "Commit" tells Git that you're making a commit. "-m" means message which tells git that the following line in quotation marks is your commit message. I'll explain commits later on, but for now, just explain what you changed or uploaded.

```
C:\Users\limri\Documents\GitHub Tutorials>git add .

C:\Users\limri\Documents\GitHub Tutorials>git commit -m "Add files ABC057D.java, ABC05D.java, and ABC114D.java"
[master (root-commit) fee919a] Add files ABC057D.java, ABC05D.java, and ABC114D.java
 3 files changed, 160 insertions(+)
 create mode 100644 ABC057D.java
 create mode 100644 ABC085D.java
 create mode 100644 ABC114D.java
```

Here's what it looks like in my terminal!

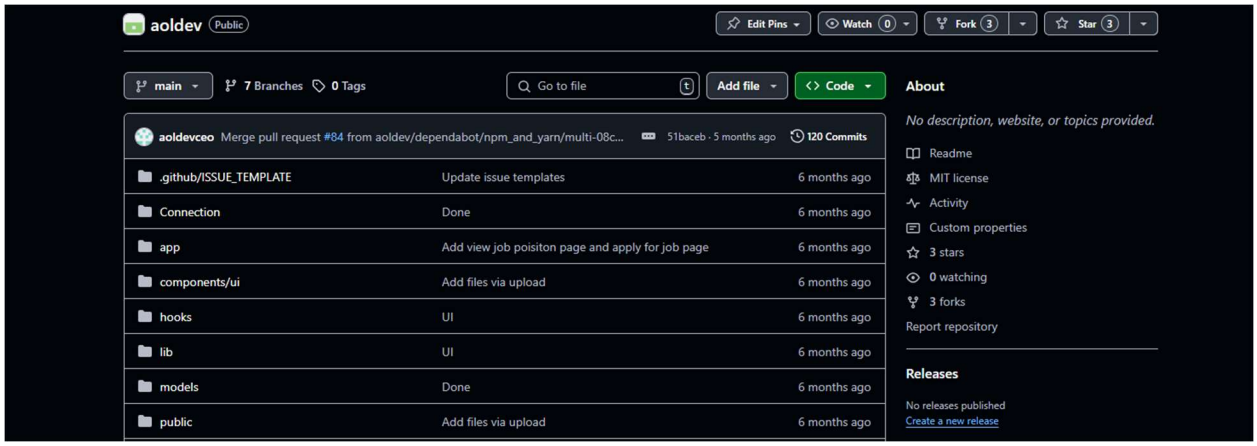
Copy the commands under the section that says "…or push an existing repository from the command line" (this is what we're doing!) and run them in your terminal.

```
git remote add origin <your repository's link>
git push origin -u master
```

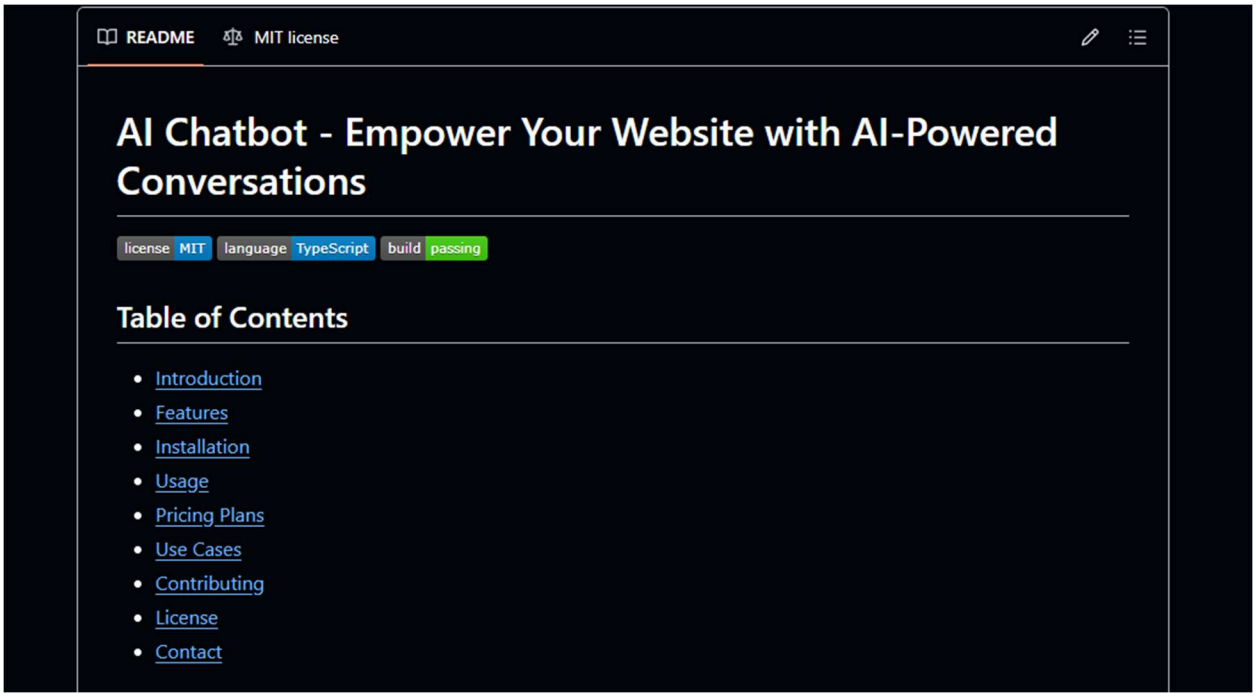
```
C:\Users\limri\Documents\GitHub Tutorials>git remote add origin https://github.com/rairayy/github-tutorial-2.git

C:\Users\limri\Documents\GitHub Tutorials>git push -u origin master
fatal: HttpRequestException encountered.
   An error occurred while sending the request.
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.97 KiB | 1.97 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/rairayy/github-tutorial-2.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Here's what it looks like in my terminal!

If you go back to your repository on GitHub, your latest commit be there!

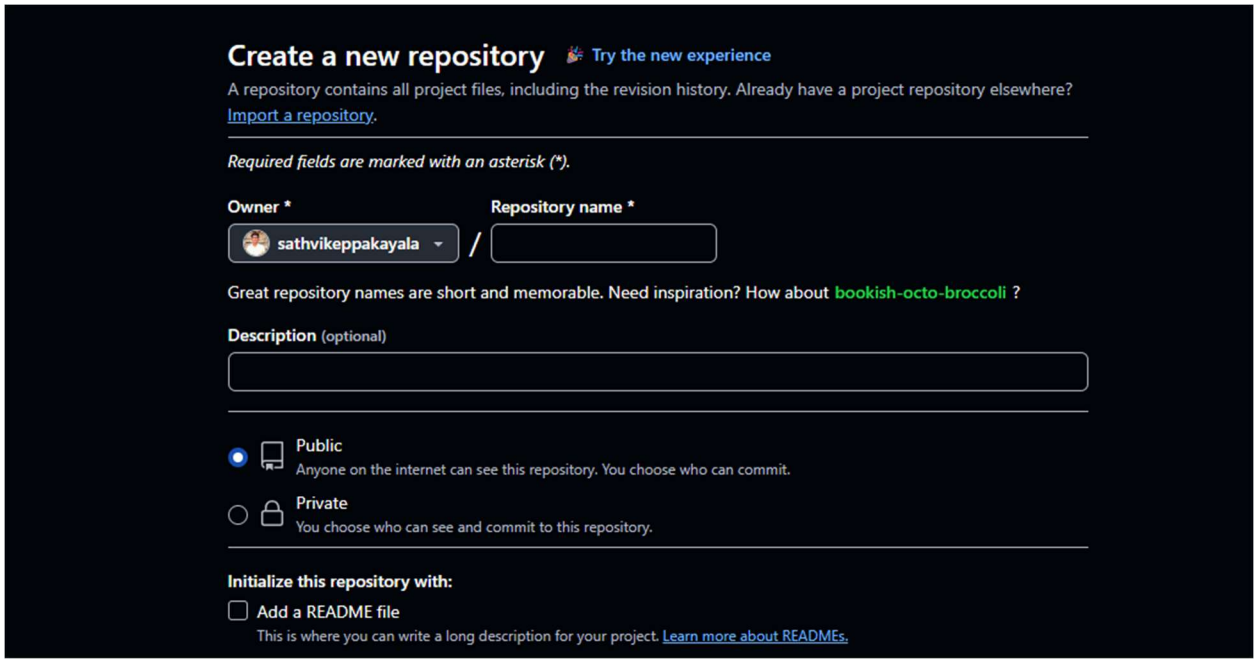


GitHub will also suggest that you add a README to your project. It'll look something like this when you've made one. When people visit your repository, the README is what they'll see right after your files so it's best to explain things there such as a more detailed description of your project or what dependencies your project has. (Dependencies are things people will need to install to make your project run on their devices, aka what other technologies your project *depend* on)

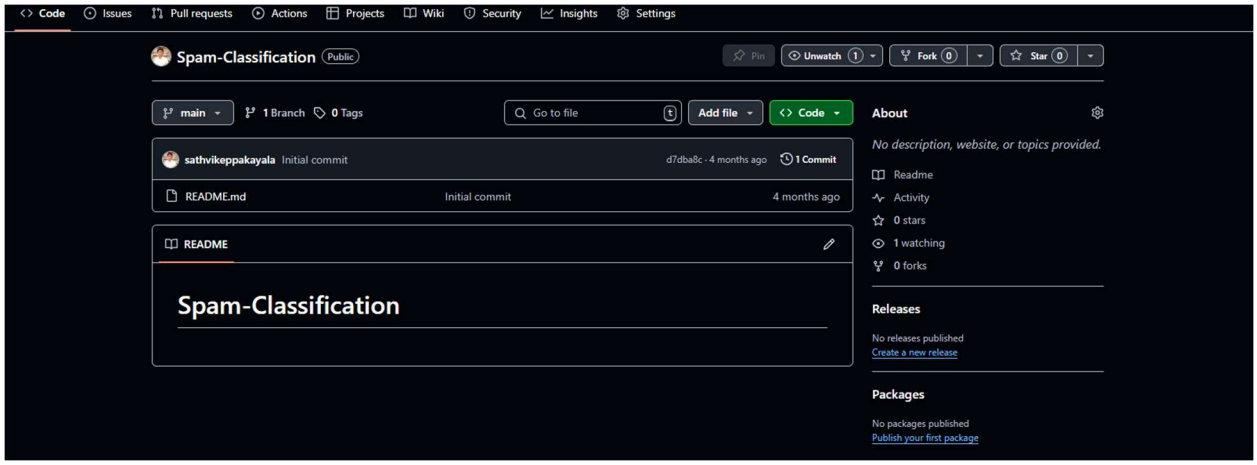


2. Starting a project in GitHub

Since you're starting a project from scratch, you can tick the box that says "Initialize this repository with a README" and create a repository.



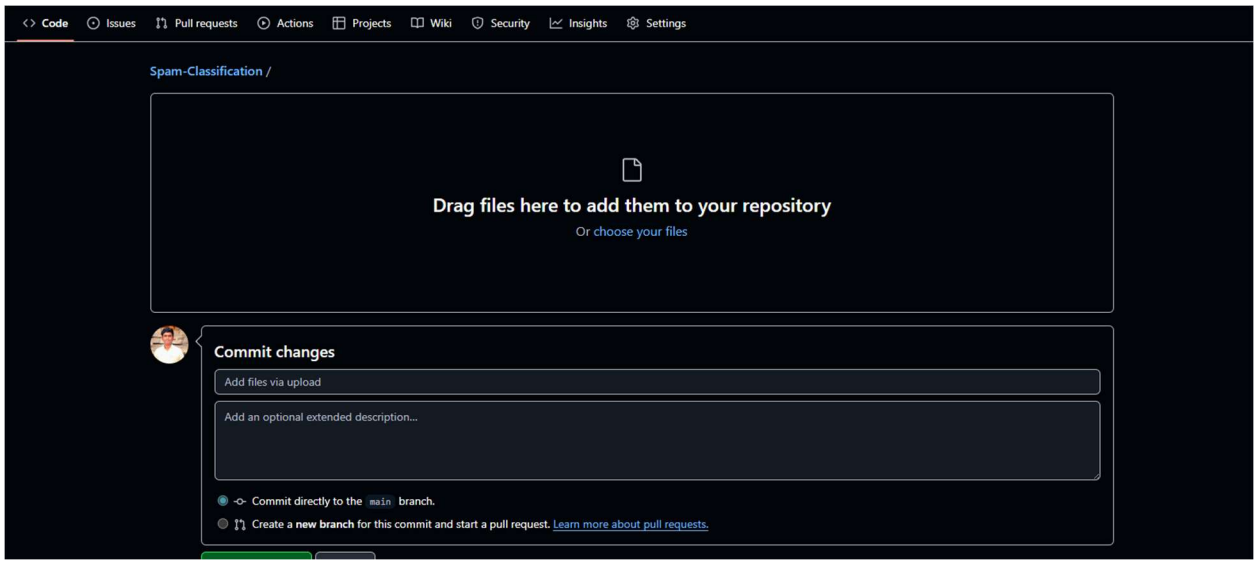
You will be taken to a page that looks like this:



If you want to upload or create files using the website, you can simply press the "Create new file" or "Upload files" buttons. Either way, you'll be asked to "Commit changes" after.

It will also let you choose between committing to the master branch or creating a new branch. You can think of branching like creating a copy of the current state of your repository, but I will explain this later on as well. If you're just using a practice repository, you can commit to the

master branch for now. **Note that we don't usually commit to the master branch, we usually make another branch first.**



If you want to work on your terminal, move to the folder you'd like to work in and run the command git init to initialize a repository.

After you've made some changes, you can run these commands which I've discussed in #1.

```
git add .
git commit -m <your commit message>
git remote add origin <your repository's link>
git push origin -u master
```